УДК 004.4

Студ. И.А. Скворцов
Науч. рук. ст. преп. А.С. Наркевич
(кафедра программной инженерии, БГТУ)

ОСОБЕННОСТИ РЕАЛИЗАЦИИ КОМПИЛЯТОРА SIA-2020

Язык программирования *SIA-2020* является Си-подобным, транслируемым в язык *assembler*.

Транслятор *SIA-2020* состоит из четырех основных фаз: лексического, синтаксического, семантического анализа и генерации кода. Первая фаза выполняется лексическим анализатором, задачей которого является замена ключевых слов, идентификаторов и литералов на лексемы (один символ) для простоты работы следующих этапов транслятора. Вторым этапом является синтаксический анализ, в его основе лежит контекстно-свободная грамматика. Задачей синтаксического анализатора является сравнение входных цепочек лексем на соответствие правилам грамматики. В результате успешной работы на выходе получаем дерево разбора. Третий этап реализуется семантическим анализатором для проверки смысловой связи между узлами дерева. В случае возникновения ошибок на любой фазе генерация кода не производится. Заключительным этапом является генерация кода в язык *assembler*.

Особенностями транслятора *SIA-2020* является поддержка четырех базовых типов данных: *byte* (1), *long* (4), *bool* (1), *string* (255); массивов на их основе; использование массивов и указателей в качестве параметра функции; реализация библиотеки, которая проверяет ошибки в ходе выполнения программы; отключаемый контроль на переполнение целочисленных переменных. В языке поддерживаются также одно- и четырех- байтовые целочисленные и булевы массивы. Допустимы операции сравнения и арифметические (бинарные и унарные), для работы со строками используются библиотечные функции. Генерация кода реализована как машина для регистро-стекового исполнения кода. При выполнении кода предпринимается попытка использовать регистры и память, что эффективнее, нежели использование только стека для всех вычислительных действий в программе. При генерации исполняемого кода операнды и результаты помещаются в регистры (без использования стека) при этом, если выражение для обработки сложное и свободных регистров не остается, тогда используется стек. Заполнение элементов массивов любого размера реализовано с использованием только регистров, как и начальная инициализация переменных. Такой принцип позволяет выполнять программу за меньшее число тактов.